Enumerating Polyominoes of p4 Tiling by the Reverse Search (Theoretical Computer Science and Its Applications)

HORIYAMA, Takashi; SAMEJIMA, Masato

数理解析研究所講究録
第1649巻 2009年 55-57

Enumerating Polyominoes of p4 Tiling by the Reverse Search
逆探索による p4 タイリングの列挙

Takashi HORIYAMA　　Masato SAMEJIMA
堀山 貴史　　鮫島 真人

Graduate School of Science and Engineering, Saitama University
埼玉大学 理工学研究科

1 Introduction

Polyominoes are the two dimensional shapes made by connecting n equal-sized squares, joined along their edges. The first pentomino (i.e, the polyomino consisting with five squares) problem appeared in 1907 by Dudeney, and the name polyomino was invented in 1953 by Golomb [4]. Since then, enumerating and counting the number of polyominoes has been of interests in combinatorial geometry, puzzle, and recreational mathematics (see, e.g., [5, 6]).

Another aspect of the interests is on polyomino tilings. Domino tiling is one of the most famous problem: Enumerate or count the number of ways to cover a given region by dominoes (or by specified polyominoes in general tilings), where neither overlapping nor hanging over the region is allowed. Enumeration and counting for covering the two dimensional plane are also considered. Recently, Fukuda et al. proposed a new tiling problem, in which p4 isohedral tilings are enumerated [2, 3]. More precisely, in the p4 isohedral tiling problem, the coordinates of the *origin* and the *terminus* (x, y) are given, where x and y are integers, and w.l.o.g. we can assume that the origin is $(0, 0)$ and $y \geq x \geq 0$. Our task is to enumerate n-ominoes, i.e., polyominoes with $n = (x^2 + y^2)/2$ squares, that cover the plane only with the rotations of 90 degrees. Since x and y are restricted to be integers, n can be 1, 2, 4, 5, 8, 9, 10, 13, 16, 17, ... Note that the parity of x always corresponds to that of y.

In this paper, we propose the use of the reverse search [1]. Under this search scheme, we first define a rooted tree whose nodes correspond to the polyominoes of p4 tilings. By designing a rule for obtaining child nodes from any node in the tree, we can enumerate all nodes (i.e., all tilings) by traversing the tree. This technique has the following two characteristics: (1) No trial and error, since we have a rule. Thus, we can reduce the computation time. (2) No need to record the already enumerated tilings. We can reduce the space complexity.

2 Family Trees and the Enumeration

First, we define a tree structure among polyominoes of p4 tiling. Let the *root polyomino* consist of the following two regions: (1) rectangle of size xy whose down-left is the origin and up-right is (x, y). (2) rectangle of size $(y-x)^2/2$ whose down-left is $(x, 0)$ and up-right is $((x+y)/2, y-x)$. In case $x = 0$, the root polyomino consists only of rectangle (2). The *neighbors* of a polyomino P is the polyominoes by deleting a square from P and adding another square to it so that the resulting polyominoes should keep the connectivity. Since the 90-degrees rotations of p4 tiling compose an equivalence relation for the deleting and adding squares [2], the deleting and adding squares belong to the same equivalence class.

The *parent* of a polyomino P is a neighbor of P in which the "level" of the adding square is smaller than that of the deleting one. We call this the *fundamental rule*. If P has two or more such squares, we break the tie by setting a total order among the equivalence classes. We call this the *tie-breaking rule*, and we assume that classes have their corresponding integers and the class of the smallest integer is the winner. The *level* of a square is the distance from the root polyomino: The squares in the root polyomino have level 0. Those next to the root polyomino have level 1. Similarly, those next to the squares in level i have level $i+1$. By repeating the process of obtaining the parent, we can observe a sequence from P to the root polyomino. By merging such sequences, we can obtain a *family tree*, in which nodes correspond to the n-ominos, and edges correspond to the pairs of n-ominos and their parents.

Now, we define the rule for obtaining children from any node in the family tree. This rule is obtained by reverting the rules from a node to its parent. The reverse of the fundamental rule is as follows: Children of a polyomino P is a neighbor of P in which the level of the adding square is larger than that of the deleting one. For avoiding to generate the same polyomino twice, we use an additional rule: If a deleted square has a candidate of its adding one of smaller level, we do not admit the squares of larger level. The reverse of the tie-breaking rule is as follows: Let s be the square added in obtaining P. (If P is the root node, we assume s as a virtual square with an integer that is larger than any of the equivalence classes.) We admit deleting/adding a square of smaller class than that of s, or deleting/adding a square so that s cannot move, i.e., any move of s destroy the connectivity.

The connectivity of a polyomino can be checked in $O(n)$ time. In some cases, the check can be done in constant time. The key observation is that if a polyomino for tiling a plane is connected then it contains no hole. By utilizing this observation, we obtain the connectivity lemma. Let P_p and P_c denote a polyomino and its child, and P_p is connected. Let s_p denote the deleted square for moving from P_p to P_c. Then, P_c is connected if one of the following three conditions holds: (1)

	o	
x	s_p	x
x	x	x

(2)

	x	
o	s_p	o
o	o	o

(3)

	x	x
o	s_p	x
o	o	

The cells o and x respectively denote the existence and nonexistence in the place. The cells without any mark denote that they can be either o or x. In other cases, we still need $O(n)$ time for checking the connectivity.

We implemented the algorithm and enumerated polyominoes of p4 tiling up to $n = 17$. The number of p4 n-ominoes for $n = 1, 2, 4, 5, 8, 9, 10, 13, 16, 17$ is 1, 2, 9, 21, 166, 317, 596, 4,152, 26,448, 48,960, respectively. Figure 1 is a partial list of enumerated 17-ominoes for p4 tiling. The computation time is small. It takes only 38.7 second even for 17-ominoes on Intel Core2 1.2 GHz CPU with Debian GNU/Linux.

References

[1] D. Avis and K. Fukuda, Reverse Search for Enumeration, *Discrete Appl. Math.*, 6, pp.21–46, 1996.

[2] H. Fukuda et al., A Method to Generate Polyominoes and Polyiamonds for Tilings with Rotational Symmetry, *Graphs and Combinatrics*, 23, pp.259-267, 2007.

[3] H. Fukuda et al., Enumeration of Polyominoes, Polyiamonds and Polyhexs for Isohedral Tilings with Rotational Symmetry, *Lecture Notes in Computer Science* 4535, pp.68-78, 2008.

[4] S. Golomb, *Polyominoes: puzzles, Patterns, Problems, and Packings*, Princeton University Press, Princeton, NJ, second edition, 1994.

[5] D. E. Knuth, `http://www-cs-faculty.stanford.edu/~knuth/programs.html`.

[6] T. R. Parkin et al., Polyomino Enumeration Results, *SIAM Fall Meeting*, California, 1967.

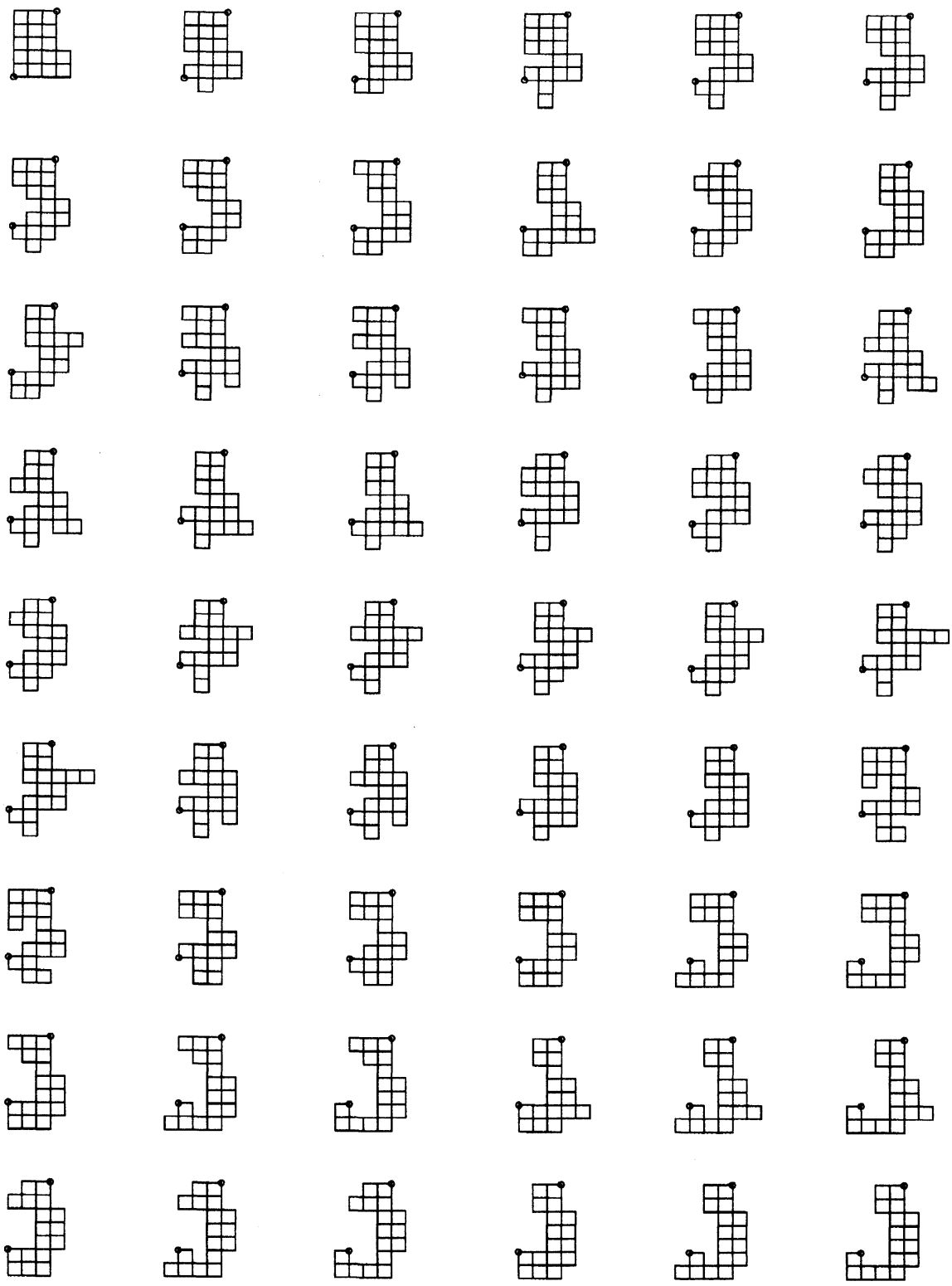

Figure 1: Partial list of 17-ominoes for p4 tiling.